WHAT IS CLAIMED IS:

1. A method for processing logic operations of a network model, comprising the steps of:

a) partitioning the logic operations in the network model into a plurality of domains, wherein a domain has a respective total number of operations;

b) ordering, in first domain orderings, the operations of the respective domains, wherein each operation has a rank and determinable numbers of operations between it and each respective one of the other logic operations of the domain;

c) identifying, in the respective domain orderings, instances of multiple operations having dependencies on respective common source operations from other ones of the domains; and

d) ordering, in second domain orderings, the operations of the respective domains, wherein pairs of the operations having the common dependencies are separated by at least as many operations as the total number of operations in the domains of the respective source operations, so that after one value is computed for one instance of an operation depending on a source operation, a next value is computed for the source operation before computing the next instance of an operation depending on the source operation.

2. The method of claim 1, wherein one of the second domain orderings includes at least one waiting operation between a pair of the logical operations of one of the first domain orderings.

3. The method of claim 2, comprising the step of:

e) ordering, in a first merged ordering, the operations of all the domains, wherein the first merged ordering is responsive to the respective domain orderings.

4. The method of claim 3, wherein the first merged ordering omits any waiting operations that are in the domain orderings.

5. The method of claim 4, comprising the step of:

f) ordering, in a second merged ordering, the operations of all the domains, wherein the second merged ordering includes separations between pairs of the operations having a common dependency, the separations being of at least the extent as the separations of step d).

6. The method of claim 5, wherein step f) comprises the step of:

comparing relative rankings in the first merged ordering of the operations having a common dependency on a source operation, to identify deficits in numbers of intervening operations separating pairs of the instances of the dependent operations; and

identifying cases of overlap in intervening operations between first and second ones of the pairs of operations having a common dependency, so that a reduced number of waiting operations are required for the second merged ordering to satisfy the deficits for both the pairs.

7. A computer program product for processing logic operations of a network model, comprising:

a) first instruction means for partitioning the logic operations in the network model into a plurality of domains, wherein a domain has a respective total number of operations;

b) second instruction means for ordering, in first domain orderings, the operations of the respective domains, wherein each operation has a rank and determinable numbers of operations between it and each respective one of the other logic operations of the domain;

c) third instruction means for identifying, in the respective domain orderings, instances of multiple operations having dependencies on respective common source operations from other ones of the domains; and

d) fourth instruction means for ordering, in second domain orderings, the operations of the respective domains, wherein pairs of the operations having the common dependencies are separated by at least as many operations as the total number of operations in the domains of the respective source operations, so that after one value is computed for one instance of an operation depending on a source operation, a next value is computed for the source operation before computing the next instance of an operation depending on the source operation.

8. The computer program product of claim 7, wherein the fourth instruction means includes instruction means for ordering one of the second domain orderings to include at least one waiting operation between a pair of the logical operations of one of the first domain orderings.

9. The computer program product of claim 8, comprising:

e) fifth instruction means for ordering, in a first merged ordering, the operations of all the domains, wherein the first merged ordering is responsive to the respective domain orderings.

10. The computer program product of claim 9, wherein the first merged ordering omits any waiting operations that are in the domain orderings.

11. The computer program product of claim 10, comprising:

f) sixth instruction means for ordering, in a second merged ordering, the operations of all the domains, wherein the second merged ordering includes separations between pairs of the operations having a common dependency, the separations being of at least the extent as the separations provided by the fourth instruction means.

12. The computer program product of claim 11, wherein the sixth instruction means comprises:

instruction means for comparing relative rankings in the first merged ordering of the operations having a common dependency on a source operation, to identify deficits in numbers of intervening operations separating pairs of the instances of the dependent operations; and

instruction means for identifying cases of overlap in intervening operations between first and second ones of the pairs of operations having a common dependency, so that a reduced number of waiting operations are required for the second merged ordering to satisfy the deficits for both the pairs.

13. An apparatus for processing logic operations of a network model, comprising:

a) means for partitioning the logic operations in the network model into a plurality of domains, wherein a domain has a respective total number of operations;

b) means for ordering, in first domain orderings, the operations of the respective domains, wherein each operation has a rank and determinable numbers of operations between it and each respective one of the other logic operations of the domain;

c) means for identifying, in the respective domain orderings, instances of multiple operations having dependencies on respective common source operations from other ones of the domains; and

d) means for ordering, in second domain orderings, the operations of the respective domains, wherein pairs of the operations having the common dependencies are separated by at least as many operations as the total number of operations in the domains of the respective source operations, so that after one value is computed for one instance of an operation depending on a source operation, a next value is computed for the source operation before computing the next instance of an operation depending on the source operation.

14. The apparatus of claim 13, wherein the means for ordering the operations in second domain orderings includes means for ordering one of the second domain orderings to include at least one waiting operation between a pair of the logical operations of one of the first domain orderings.

15. The apparatus of claim 14, comprising:

e) means for ordering, in a first merged ordering, the operations of all the domains, wherein the first merged ordering is responsive to the respective domain orderings.

16. The apparatus of claim 15, wherein the first merged ordering omits any waiting operations that are in the domain orderings.

17. The apparatus of claim 16, comprising:

f) means for ordering, in a second merged ordering, the operations of all the domains, wherein the second merged ordering includes separations between pairs of operations having a common dependency, the separations being of at least the extent as the separations provided by the means for ordering the operations in second domain orderings.

18. The apparatus of claim 17, wherein the means for ordering the operations in a second merged ordering comprises:

means for comparing relative rankings in the first merged ordering of the operations having a common dependency on a source operation, to identify deficits in numbers of intervening operations separating pairs of the instances of the dependent operations; and

means for identifying cases of overlap in intervening operations between first and second ones of the pairs of operations having a common dependency, so that a reduced number of waiting operations are required for the second merged ordering to satisfy the deficits for both the pairs.